Automatic Categorization of Software Modules

Parvinder Singh Sandhu [†] **, Madhu Bala**[††] **and Hardeep Singh**[†††]**,**

Guru Nanak Dev Engineering College (Ludhiana), Guru Nanak Dev University (Amritsar)- India

Summary
The world of software has demonstrated the remarkable appeal of communal software development. Large number of software projects can leverage, reuse, and coordinate their work through internet and web-based technology. For example, Source-Forge currently hosts about sixty thousand software systems, similar strategies have suggested for corporate software development. With thousands of projects, manually locating related projects can be difficult. Hence to use automatic software categorization to find clusters of related software projects using only the source code from projects, automatic categorization of software experiments with a set of programs. Automatic categorization of software systems is a novel and intriguing challenge on software archive. Evolution has focused on determining intra-component relations of given software system also increase to differentiate between categories. Function oriented produces better result than the object oriented. Automatic categorization of software has provided better results than LSA retrieval techniques in terms of Precision and Recall with multinomial Naïve Bayes scheme has outperformed all other approaches and shows better results than the existing approach (SVD), being used by some open source code repositories e.g. Source forge Hence, the tool can also be utilized for the automatic categorization of software components and this kind of automation may improve.

Key words:
LSA, SVD, Machine Learning.

1. Introduction

Manual categorization generally requires deep understanding of not only the target software system, but also other software systems and their classification policy. With the increase in the number of software systems, e.g., Source-Forge now has over fifty-five thousand software systems registered and continues to evolve, such manual identification is not enough It is important for software evolution to search and use existing similar software systems from software archive. An evolution history of an existing similar software system is useful. It may even evolve a software system based on an existing one. Automatic software categorization algorithm used to help to finding similar software systems in software archive.
Categorization and explore several known approaches including code clones-based similarity metric, decision trees, and latent semantic analysis. Automatic categorization would be helpful in several ways [1].

I) Several of similar software can be group together in a category for Ease of browsing

II) Developers working on a software system May informed about related Software. Developers can learn from experience with such Software Systems. They can avoid duplicate work and promote more Reuse.

The accuracy of all automatic categorization systems is highly dependent upon the effort and care taken during the training or rule definition phase As the number of categories increases, the number and complexity of the rules must also increase to differentiate between categories [2].

A classification strategy is presented that involves the use of supervised and unsupervised pattern classification and multivariate visualization. These Techniques are applied to the profiles executions in order to group together failures with the same or similar causes. Classification algorithms grouped into supervised and unsupervised methods, although some algorithms combine features from each group [3].

When used in a corporate setting, infrastructures for project information sharing present new Opportunity. For example, all projects that Have something in common, so that the project Groups can collaborate and share their work. With thousands of projects, manually locating related projects can be difficult. Hence, use of Automatic software categorization to find group of related software projects, using only the source Code from projects. Potential for automatic Categorization of software systems without human

Kawaguchi in [1] used code clones-based similarity metric, decision trees, and latent semantic analysis (LSA) approaches to help finding similar software systems in software archive. Further, Kawaguchi in [2] explained the use of LSA approach to automatic categorization of software systems and developed web interface to visualize determined categories.

Latent Semantic Analysis (LSA) using Singular Value Decomposition (SVD). The purely and hybrid Naïve Bayes schemes are also tried for the Categorization of the software components.

Manuscript received August 5, 2007
Manuscript revised August 20, 2007

2. Methodology Followed

The current study is based on Object Oriented and Function Oriented techniques. In this paper, the present work i.e. machine learning algorithms based technique is compared with the already existing techniques i.e. LSA to find out the best technique. The Naive Bayesian classification is the optimal method of supervised learning.
In this paper there are six types of machine learning algorithm i.e. Bayes, lazy, Meta Learner, Misc, Trees, goal is to best algorithm for five-categories i.e. Graphics and Games, File Input/Output, Network Application, Driver Hardware, Memory Oriented. After finding the best algorithm, compare this algorithm with Linguistic techniques. There are certain algorithms present in all the classes which are analyzed individually to find the over all algorithm The categorization results are close to the manual analysis, used to perform by the repository managers. Hence the developed tool can be also be utilized for the automatic evaluation phase statistics. Categorization of software components and domain relevancy of software components. This kind of automation may improve the productivity and quality of software development.
The present approach followed is performed over Multinomial Naive Bayes is fast, easy to Implement and relatively effective. The multinomial naive bayes there is modeling and classification. In the Multinomial Naive Bayes models the distribution of words in a document as a multinomial. A document treated as a sequence of words and it is assume that each word position generated independently of every other. This is easy to see for binary classification, where the boundary is defined by setting the differences between the positive and negative class parameters equal to zero. The multi-variety performs well with small vocabulary sizes, but that the multinomia performs usually performs even better at larger vocabularysizes[12]. Various steps used for automatic categorization of software.
(i) Collection of Data: Data in the form of programs are collected, the various programs collected basically belongs to both object oriented (object is taken as the vital entity) and function oriented techniques (functions only take inputs and produce outputs, and don't have any internal state that affects the output produced for a given input), are divided into following categories for each. i.e. Graphics and Games, File Input/Output, Network Application, Driver Hardware, Memory Oriented.

ii) Keyword Extraction: The basic on the extraction of words, this is extract those names from the data which are relevant in words of giving out the basic functioning of a particular program, in order to obtain this all the keywords which are not of any relevance should be excluded from the current context.
(iii) Preprocessing of Data for Further Analysis: From the frequency matrix is used and a file is created for further analysis. Basically there are following formats in which file can be converted and used for further analysis i.e ARFF. ARFF file converts the output from the previous steps to the relevant usable file default.
(iv) Evaluation of preprocessed data: The arff generated in previous step is taken and various techniques are applied to it, it can note that Files are generated e for the Function Oriented Data. 10 fold cross validation is used in the performance of each learning algorithm was evaluated using 10 complete runs of 10-fold cross-validation. Cross-validation this means that 100 calls of one classifier with the training data and tested against test data. In each 10-fold cross-validation, each data set randomly split into 10 equal-size segments and results averaged over 10 trials.
(v) Evaluation of Results and Selection of best Algorithm**:** From the above each algorithm produce results those results are collected for function based approach. From the results best technique for the categorization is extracted and over all best technique is presented which can be categorized very easily and accurately.

- Use training set: The classifier is evaluated on how well it predicts the class of the instances it trained
- Cross-validation: The classifier evaluate by cross-validation, using the numbers of folds that entered in the folds text field.

(vi) Approach wise best Algorithm: It is to be noted that all the above 5 steps are performed for both function oriented and object oriented approach. Afterwards comparison is performed to get the best Approach.
(vii) Comparison with existing techniques: The results extracted from the previous steps are compared with the existing techniques i.e. LSA. LSA is performed over all the input files and the results extracted from these are compared with the present work.
The Latent Semantic Analysis (LSA) can be applied to induce and represent aspects of the meaning of English language words. LSA is a variant of the vector space model that converts a representative sample of documents to a term-by-document matrix in which each cell indicates the frequency with which each term (rows) occurs in each document (columns). Thus, a document becomes a column vector and can be compared with a user's query represented as a vector of the same dimension.

In SVD based technique, the query component's similarity with the other components in the repository is measured by calculating the cosine between the vectors, xk and a query vector, qk as shown below:

$$S = \tilde{q}^T \tilde{A} \quad (4)$$

Where

$$\tilde{A} = S_K^{1-\alpha} P_K^T \quad (5)$$

And the query vector is projected into the same *k*-dimensional space [20] by:

$$\tilde{q} = q^T W_K S_K^{\alpha} \quad (6)$$

The performance of queries generally improves as *k* increases, but will decrease past a threshold. It is possible for an SVD based system to locate terms, which do not even appear in a document. A document that is located in a similar part of the Precision recall data for SVD concept space (i.e. which have a similar meaning) is retrieved, rather than only matching keywords.

2.1 Evaluation of Developed System

It is tried to evaluate the system in terms of Precision and Recall criteria. Let *S* be a set of all software systems contained in a repository. Precision and Recall are defined in (18).

$$Precision = \frac{\sum_{s \varepsilon S} precision_{soft}(s)}{|S|} \quad (7)$$

Where

$$precision_{soft}(s) = \frac{|C_{Actual}(s) \cap C_{Ideal}(s)|}{|C_{Actual}(s)|} \quad (8)$$

And

$$Recall = \frac{\sum_{s \varepsilon S} recall_{soft}(s)}{|S|} \quad (9)$$

where

$$recall_{soft}(s) = \frac{|C_{Actual}(s) \cap C_{Ideal}(s)|}{|C_{Ideal}(s)|} \quad (10)$$

Where $C_{actual}(s)$ is a set of clusters containing software "s", generated by our software and $C_{Ideal}(s)$ is a set of clusters containing input software "s", determined manually by the domain experts. Using Precision and Recall values F-value is calculated as a measure of performance evaluation i.e.

$$F\text{–}Value = \frac{2pr}{p+r} \quad (14)$$

Where *p* is the Precision and *r* is the Recall of the system.

3. Implementation and Results

As a software implementation of the discussed concept, a deployable function oriented based component, Which Microsoft's binary standard is for object interoperability is developed. The developed component's objects can be accessible through C++ or any other language that supports function oriented or object oriented. A sample data from various Reusable Repositories of 'C++' components is collected belonging to five categories

3.1 Selection of best approach in Functional oriented:

In this section, the various selected algorithms based on maximum accuracy and minimum error are shown in Table 1.1. The basis of the comparison is the 10-fold statistics, on the first term on which comparison is carried out is accuracy in case some of the algorithms have same accuracy then the comparison is performed on the basis of least error amongst the algorithms being compared. The thorough analysis of each algorithm selected is given below :

The first algorithm selected in Bayes class is multinomial Naïve Bayes with 72.9167 (Accuracy %),0.107(MAE), 0.3187 (RMSE), which is incomparable with the other members of the class except complement naïve bayes its accuracy is less

- Second algorithm selected from Function Class is Simple logistic with 47.917(Accuracy %), 0.2083(MAE), 0.4564 (RMSE), as it has highest accuracy amongst the another functional class
- Third algorithm selected from the Lazy class is LWL[33][34] with 43.75 (Accuracy %), 0.2435 (MAE), 0.3642 (RMSE), as it has highest accuracy amongst the class
- Fourth algorithm selected from Meta Class is Logic Boosting [35][36] with 66.6667(Accuracy %), 0.1352 (MAE), 0.3036 (RMSE), as it has highest accuracy amongst the class.
- Fifth algorithm selected from the Misc Class is VFI [37] with 58.3333 (Accuracy %), 0.2319 (MAE), 0.3749 (RMSE), as it has highest accuracy amongst the class.

Finally, Sixth algorithm selected from Trees class is LMT [38] with 66.6667 (Accuracy %), 0.1293 (MAE), 0.3262 (RMSE), as it has highest accuracy amongst the class.

Table 1. Classification Results of best algorithms used for Function Oriented Software Data

Algorithm	Classification Training Statistics			Statistics after 10 fold Cross-Validation		
	Accuracy (%)	MAE	RMSE	Accuracy (%)	MAE	RMSE
NaïveBayes Multinomial	95.8333	0.0167	0.1291	72.9167	0.107	0.3187
Simple logistic	100	0	0	47.9177	0.2083	0.4564
LWL	66.6667	0.2113	0.3151	43.75	0.2435	0.3642
Logic boosting	100	0.0024	0.0052	66.6667	0.1352	0.3036
Mean VFI	89.5833	0.3175	0.3968	58.333	0.2319	0.3749
LMT	100	0	0	66.6667	0.1293	0.3262

Table 2. Classification Results of best algorithms used for the Object Oriented Software Data

Algorithm	Classification Training Statistics			Statistics after 10 fold Cross-Validation		
	Accuracy (%)	MAE	RMSE	Accuracy (%)	MAE	RMSE
Complement naïve bayes	91.0448	0.0358	0.1893	55.2239	0.1791	0.4232
RBF Network	95.5224	0.032	0.1264	54.2239	0.1818	0.417
IB1	98.5075	0.006	0.0773	32.8358	0.2687	0.4183
Random committee	98.5075	0.006	0.0546	49.2537	0.2487	0.3715
VFI	83.5821	0.3187	0.3983	44.7761	0.2648	0.4121
Random tree	98.5075	0.1318	0.1318	37.3134	0.2647	0.3809

3.2 Selection of best approach for OO Data

In this section, the various selected algorithms based on maximum accuracy and minimum error are shown in Table 2.

- Bayes class is complement naïve bayes with 55.2239 (Accuracy %), 0.1791(MAE), 0.423(RMSE), which is incomparable with the other members of the class except naïve bayes multinomial its accuracy is less.
- Second algorithm selected from Function Class is RBF networks with 54.223(Accuracy %), 0.1818(MAE), 0.417 (RMSE), as it has highest accuracy amongst the another functional class.
- Third algorithm selected from the Lazy class is IB1 with 32.8358(accuracy %), 0.2687 (MAE), 0.4183(RMSE), as it has highest accuracy amongst the class.
- Fourth algorithm selected from Random committee with 49.2537(Accuracy %), 0.2487 (MAE), 0.3715 (RMSE), as it has highest accuracy amongst the class.
- Fifth algorithm selected from the Misc Class is VFI with 44.7761(Accuracy %), 0.2648(MAE), 0.4121(RMSE), as it has highest accuracy amongst the class.
- Finally, Sixth algorithm selected from Trees class is Random tree with 37.3134 (Accuracy %), 0.1293 (MAE), 0.3262 (RMSE), as it has highest accuracy amongst the class.

The accuracy of best algorithm in object oriented. i.e. complement naïve bayes is 55.2239%, and the accuracy of best algorithm in functional oriented i.e. multinomial naïve bayes is 72.9167 %. A result shows accuracy of multinomial is high.

Table 3. Result of SVD by using precision, recall and F-measure

Precision	Recall	F-Measure
1.0000	0.2500	0.4000
0.1500	0.4286	0.2222
0.8000	0.5000	0.6154
0	0	0
0.2500	0.3846	0.3030

If SVD and multinomial are compared on the basis of precision, Recall and F-measure As shown in table 1.3 and 1.4 multinomial performs better results than the existing approach.

Table 1.3: Result of Multinomial by using precision, recall and F-measure

Precision	Recall	F-Measure
0. 0.6	0.667	0.632
0.625	0.714	0.667
0.833	0.625	0.714
0.909	0.909	0.909
0.692	0.692	0.692

4. Conclusion

The results on the basis of accuracy of functional oriented and object oriented, function oriented performs better than the object oriented. The multinomial naïve based software categorization approach provides better results than purely SVD based retrieval techniques in terms of Precision and Recall. The categorization results are close to the manual analysis, used to be performed by the programmers/repository managers. Ultimately, this kind of automation may improve the productivity and quality of software development.

References

[1] Kawaguchi, S., P. K. Garg, M. Matsushita and K. Inoue, 2003. Automatic categorization algorithm for evolvable software archive Software Evolution. Sixth International Workshop on Principles of Software Evolution (IWPSE'03), pp. 195 – 200.

[2] Kawaguchi, S., P. K. Garg, M. Matsushita and K. Inoue, 2004. MUDABlue: an automatic categorization system for open source repositories. 11th Asia-Pacific Software Engineering Conference (2004), pp. 184 – 193.

[3] Bouckaert. R. R. (2004), "Bayesian Network Classifiers in Weka", 2004.

[4] Dinkelacker, J., Garg, P., Nelson, D. and R. Miller. (2002), "Progressive Open Source", In Proceedings of the International Conference on Software Engineering, Orlando, Florida (2002) , pp. 177-184.

[5] Frank, E., Hall, M. and Pfahringer, B. (2003), "Locally Weighted Naive Bayes", Proceedings Nineteenth Conference in Uncertainty in Artificial Intelligence, 2003, pp. 249-256.

[6] Frank, E. and Hall, M. (2001), "A Simple Approach to Ordinal Classification", Proceedings Twelfth European Conference on Machine Learning, 2001, pp. 145- 156.

[7] Ho, T. K. (1998), "The Random Subspace Method for Constructing Decision Forests", Proceeding IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, 8, 1998, pp. 832-844.

[8] Joachims, T. (1998), "Text Categorization with Support Vector Machines: Learning with Many Relevant Features". (1998)

[9] Kaariainen. M and Malinen. T. (2004), "Selective Rademacher Penalization and Reduced Error Pruning of Decision Trees", Journal of Machine Learning Research 5 (2004), pp. 1107– 1126.

[10] Rennie, J.D.M., Shih, L. (2003), "Tackling the Poor Assumptions of Naive Bayes Text Classifiers", Proceedings Twentieth international Conference on Machine Learning (ICML), 2003, pp. 616-623.

[11] WEKA, www.cs.waikato.ac.nz/~ml/weka/

[12] Yang. Y and Liu.X (1999), "A reexamination of text categorization methods", Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval 1999, pp. 42—49.

[13] McCallum. A and Nigam. K.(1998), "A Comparison of Event Models for Naive Bayes Text Classification", In AAAI-98 Workshop on Learning for Text Categorization, 1998, pp. 250-255.

[14] Pedersen. T. (2002), "A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation", Proceedings of the ACL-02 workshop on Word sense disambiguation.

[15] Freund, Y. and Schapire, R.E. (1996), "Experiments with a new boosting algorithm", Proceedings Thirteenth I international Conference on Machine Learning, San Francisco, 1996, pp. 148-156.

[16] Holte, R. C. (1993), "Very Simple classification Rules Perform Well on Most Commonly Used Datasets", Machine Learning, Vol. 11, 1993, pp. 63- 91.

[17] Shi, H. (2007), "Best-first decision tree learning", Hamilton, NZ.

[18] Ting, K. M. and Witten, I. H. (1997), "Stacking Bagged and Dagged Models", In Fourteenth international Conference on Machine Learning, San Francisco, CA, 1997, pp. 367-375.

[19] Wang. Z. and Webb. G .I (2000), "A Heuristic Lazy Bayesian Rule Algorithm", The Australasian Data Mining Workshop, 2002.

[20] Zou, B., Ma, X., Kemme, B., Newton, G . and Precup, D. (2006), "Data mining using Relational Database Management Systems", Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2006.

[21] Quinlan, J. R. (1993), "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers. [12] Yang. Y and Liu.X (1999), "A reexamination of text categorization methods", *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval 1999*, pp. 42—49.

Parvinder S. Sandhu is working as Assistant Professor in the Department of Computer Science and Engineering with Guru Nanak Dev Engineering College, Ludhiana (Punjab). He is Master of Engineering in Software Engineering, M.B.A. and Bachelor in Computer Engineering from NIT, Kurukshetra. He is doing research work leading to Ph.D. with Guru Nanak Dev University, Amritsar. He has published 11 research papers in referred International journals and 15 papers in renowned international conferences. His current research interests are Software Reusability, Software Maintenance and Machine Learning.

Hardeep Singh is working as Professor in the Department of Computer Science and Engineering with Guru Nanak Dev University, Amritsar, India. His date of birth is Feb. 16 and he has got twenty years of teaching experience. He is member of high profile committees of Government of India related with the technical education. He is Doctorate in Modeling and Design of Software Metrics for Object Oriented Systems. He has thirty five International and National publications to his name. He is live interest in Software Engineering, Object Oriented Paradigm,

Management Information & Decision Support Systems, Ergonomics, Computer Networks, Artificial Intelligence.

Madhu Bala is currenly doing her Masters from Guru Nanak Dev Engineering College, Ludhiana and doing research on the categorization of components in the software repositories under supervision of Prof. Parvinder S. Sandhu. She did her Bachelor in Computer Science and Engineeirng from Punjab technical university Jalandhar.